# MIPS Pipelined processor

## Presented by


| Name | section | BN | Code |
|---|---|---|---|
| Amir Anwar | 1 | 15 | 9220166 |
| Peter Safwat | 1 | 16 | 9220211 |
| Bishoy Wadea | 1 | 17 | 9220212 |
| Sarah Ahdy | 1 | 24 | 9220350 |


## Presented to

Dr. Mostafa Mahmoud

Eng. Mohammed Abdallah

# Table of Contents

# Instruction format of your design

For better visualization refer to this excel sheet

## Instruction code format:

| Part | OP code | Rdst | R1 | R2 | INDX | IMM |
|---|---|---|---|---|---|---|
| Number of bits | 5 | 3 | 3 | 3 | 2 | 16 |

## OP code table:

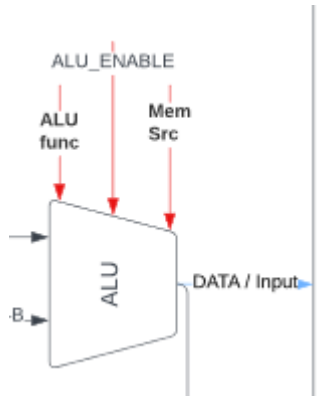
| Operation Name | Operation Code | Rdst | R1 | R2 | INDX | IMM |
|---|---|---|---|---|---|---|
| NOP | 00000 | xxx | xxx | xxx | xx | -- |
| HLT | 00001 | xxx | xxx | xxx | xx | -- |
| SETC | 00010 | xxx | xxx | xxx | xx | -- |
| RET | 00011 | xxx | xxx | xxx | xx | -- |
| RTI | 00100 | xxx | xxx | xxx | xx | -- |
| OUT Rs1 | 00101 | addr | xxx | xxx | xx | -- |
| IN Rdst | 00110 | addr | xxx | xxx | xx | -- |
| PUSH rs1 | 00111 | addr | xxx | xxx | xx | -- |
| POP Rdst | 01000 | addr | xxx | xxx | xx | -- |
| JZ Rs1 | 01001 | addr | xxx | xxx | xx | -- |
| JN Rs1 | 01010 | addr | xxx | xxx | xx | -- |
| JC Rs1 | 01011 | addr | xxx | xxx | xx | -- |
| JMP Rs1 | 01100 | addr | xxx | xxx | xx | -- |
| CALL Rs1 | 01101 | addr | xxx | xxx | xx | -- |
| INT index | 01110 | xxx | xxx | xxx | 0 or 2 | -- |
| NOT Rdst, Rs1 | 01111 | addr | addr | xxx | xx | -- |
| INC Rdst, Rs1 | 10000 | addr | addr | xxx | xx | -- |
| MOV Rdst, Rs1 | 10001 | addr | addr | xxx | xx | -- |
| ADD Rdst, Rs1, Rs2 | 10011 | addr | addr | addr | xx | -- |
| SUB Rdst, Rs1, Rs2 | 10100 | addr | addr | addr | xx | -- |
| AND Rdst, Rs1, Rs2 | 10101 | addr | addr | addr | xx | -- |
| IADD Rdst, Rs1, IMM | 11000 | addr | addr | xxx | xx | -- |
| LDD Rdst, Rs1, IMM | 11001 | addr | addr | xxx | xx | -- |
| STD | 11010 | addr | addr | xxx | xx | -- |
| LDM Rdst, IMM | 11011 | addr | xxx | xxx | xx | -- |

# Schematic diagram of the processor with data flow details.

For better understanding and better quality the original design in this link Lucid Chart

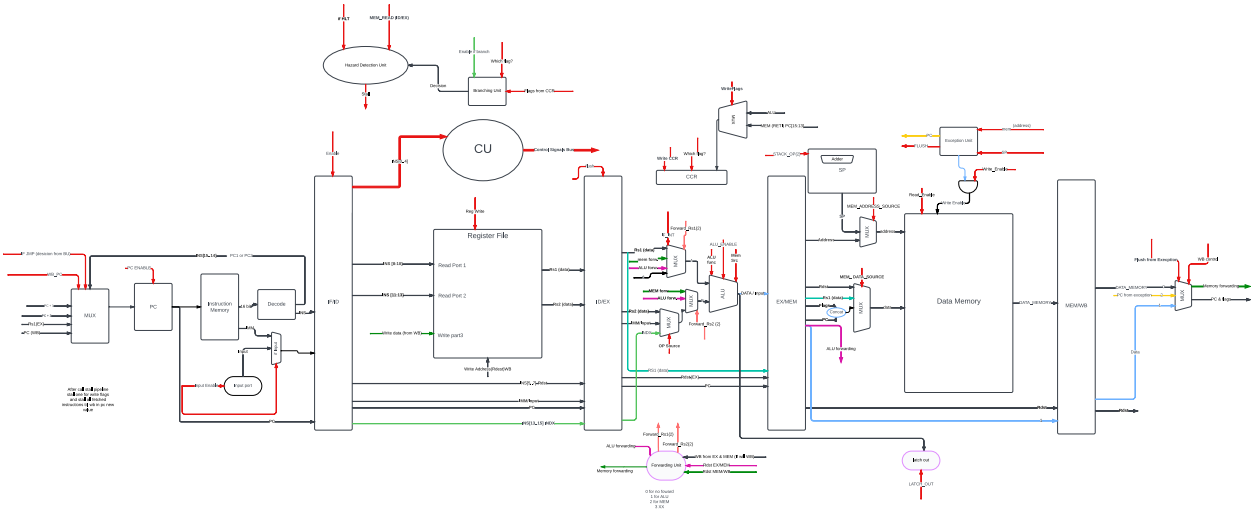
## ALU / Registers / Memory Blocks

### ALU




### Memory Blocks

## Dataflow Interconnections between Blocks & its sizes




# Pipeline stages design

## Pipeline registers details

### IF/ID Buffer:

| | |
|---|---|
| INS | 16 |
| IMM | 16 |
| PC | 16 |

### ID/EX Buffer:

| | |
|---|---|
| Rs1 | 16 |
| Rs2 | 16 |
| PC | 16 |
| Rdst | 16 |
| IMM | 16 |
| INDX | 2 |
| OP_SOURCE | 2 |
| ALU_FUNC | 3 |
| Memory_Read | 1 |
| Memory_Write | 1 |
| ALU_Enable | 1 |
| WB | 1 |
| WB_PC | 1 |
| WHICH_FLAG | 3 |
| MEM_ADDRESS_SOURCE | 1 |
| MEM_DATA_SOURCE | 2 |
| ENABLE_IF_BRANCH | 1 |
| WRITE_FLAGS | 1 |
| LATCH_OUT | 1 |
| WRITE_CCR | 1 |

| | |
|---|---|
| STACK_OP | 2 |
| WB_CONTROL | 1 |

### EX/MEM Buffer:

| | |
|---|---|
| DATA | 16 |
| PC | 16 |
| Rdst | 16 |
| Rs1 | 16 |
| Memory_Read | 1 |
| Memory_Write | 1 |
| WB | 1 |
| WB_PC | 1 |
| MEM_ADDRESS_SOURCE | 1 |
| MEM_DATA_SOURCE | 2 |
| WRITE_FLAGS | 1 |
| LATCH_OUT | 1 |
| WRITE_CCR | 1 |
| Write_OUT | 1 |
| STACK_OP | 2 |
| WB_CONTROL | 1 |

### MEM/WB Buffer:

| | |
|---|---|
| DATA_EXECUTION | 16 |
| DATA_MEMORY | 16 |
| Rdst | 16 |
| Rs1 | 16 |
| WB | 1 |
| WB_PC | 1 |
| PC_EXEPTION | 16 |
| WRITE_FLAGS | 1 |
| LATCH_OUT | 1 |
| WRITE_CCR | 1 |
| WB_CONTROL | 1 |

## Pipeline hazards and your solution

### Structural Hazards

- We solved the structural hazards by separating the data memory from instructions memory
- Making the writes with the +ve clock edge and the reads with the -ve clock edge

### Data Hazards

- We implemented Full forwarding logic (Memory-ALU & ALU-ALU)

## Control Hazards

- We take the approach of not taking the branch jump because this will improve the performance by saving 1 clk cycle in case we predict it wrong over the always taking the jump
- The decision is evaluated in the execute stage and if the result is different from the prediction then we flushed the previous 2 stage of the pipeline